CAIM
Session 6: MongoDB and MapReduce

Octavi Pascual
Lluc Bové

November 22, 2016

1 Introduction

MapReduce is a programming model for processing large quantities of data on a distributed way. MongoDB is a NoSQL database that uses storage based on JSON documents. We use MongoDB with MapReduce to write a program that finds association rules in a shopping list. This might be useful for a supermarket since associations between items exist: for example, if a client buys flour and casting sugar, it is likely that he also buys eggs because they are planning on baking a cake. This is very valuable specially for the business decision-making department[1].

2 Implementation

To solve the market basket analysis problem, we used MongoDB to store and retrieve all the data that was provided to us. Each line of the market basket data is a transaction and a transaction is composed of many items. In this context items are food products.We based our implementation on the sample code that was given to us in the session. For simplicity, we put everything in a single file *groceries.py*.

To sum up what we did, we first read the *groceries.csv* file and then we inserted the data in MongoDB. Then we used MapReduce paradigm to obtain in an efficient way all the information that we needed. That was the interesting part of the program since we had to program map and reduce procedures. Finally, we used those results to find association rules.

To count in how many transactions each item appears we simply emitted the item as key and a value of 1. The reduce procedure just sums up the values of each item, so we find the total count of the item in all the transactions.

Counting in how many transactions pairs of items appear is more interesting: we must ensure that we only emit each pair of the transaction once. For example, if the transaction is *item1, item2, item3* we must only emit three values: *(item1, item2), (item1, item3), (item2, item3)*. To do that, we just do two nested for loops. There is yet another problem to take into account. Imagine we have the following two transactions: *transaction1: item1, item2* and *transaction 2: item2, item1*. If we generate *(item1, item2), (item2, item1)* we will split the total count of the pairs. We could take that into account and search both keys in the database but it would use twice as much space. A more clever approach is to always put the lexicographically smaller item first and the other second. That is why in the map procedure we check this and we emit the values accordingly. As a side note, this problem described does not appear in the *groceries.cs* file but it is likely that in a random dataset it could appear.

3 Results

We executed our program with different values of support and confidence and we observed the number of association rules. Table 1 shows the results obtained.

Row	Support	Confidence	Nr. of association rules
1	1%	1%	426
2	1%	25%	96
3	1%	50%	0
4	1%	75%	0
5	5%	25%	4
6	7%	25%	2
7	20%	25%	0
8	50%	25%	0

Table 1: Association rules given the percentage of support and confidence

We also analysed the association rules of rows 4, 5 and 6, and they are the following:

- **4:** This row has no associations
- **5:**
  - other vegetables $\leftrightarrow$ whole milk
  - rools/buns $\rightarrow$ whole milk
  - yogurt $\rightarrow$ whole milk
- **6:**
  - other vegetables $\leftrightarrow$ whole milk

4 Conclusion

In this session we used MongoDB and MapReduce to solve a basket market analysis problem. It is interesting to think about how it would scale in a real environment. We saw that MapReduce operations were pretty fast: it took about 2 seconds to compute the total count of 9000 pairs of items. However the operations that were expensive in time were finding the association rules. This makes sense since we do those operations sequentially, iterating over each pair and computing its support and confidence. We could think about whether or not we could find a way to also use MapReduce functions to speed up the process.

References

[1] Market basket analysis: identifying products and content that go well together
http://snowplowanalytics.com/guides/recipes/catalog-analytics/market-basket-analysis-identifying-products-that-sell-well-together.html